altURI – a Thin Middleware for Simulated Robot Vision Applications

Mark N R Smith, Marwan Shaker, Shigang Yue, Tom Duckett
Centre for Vision and Robotics Research
University of Lincoln
Lincoln, UK
e-mail: mnsmith@lincoln.ac.uk

Abstract— **Fast software performance is often the focus when developing real-time vision-based control applications for robot simulators. In this paper we have developed a thin, high performance middleware for USARSim and other simulators designed for real-time vision-based control applications. It includes a fast image server providing images in OpenCV, Matlab or web formats and a simple command/sensor processor. The interface has been tested in USARSim with an Unmanned Aerial Vehicle using two control applications; landing using a reinforcement learning algorithm and altitude control using elementary motion detection. The middleware has been found to be fast enough to control the flying robot as well as very easy to set up and use.**

Keywords-middleware; robot; vision; USARSim

I. INTRODUCTION

This work is concerned with developing real-time vision based control applications for a long term project using simulated and real flying robots. This led to requirements for a suitable simulator for flying robots, and a high performance middleware with a 'constant' interface suitable for long term work. The 'environment' for the controlling application, in terms of hardware and software, was also required to be easily available for the foreseeable future.

Using a simulation environment for developing robot control applications has many advantages over using a real robot. Craighead [6] surveys many of the currently available simulator environments for unmanned vehicles and, building on previous work, suggests criteria for selecting a simulator; physical fidelity, functional fidelity, ease of development, and cost.

In the long term the overall 'product life' of a simulator and associated software becomes a factor, particularly for free, typically Open Source, simulation environments. There are several examples of free simulators that are no longer active or have become a commercial product [16] [20] [21].

The existing simulators and middleware applications were examined as candidates for use in developing real-time vision based control applications and they were found to have some important disadvantages. The existing middleware was either too closely tied to a particular, sometimes unsuitable, simulator or had significant performance or computing environment implications.

To achieve the required performance and remove the dependency on a particular simulator a new middleware toolset has been developed; altURI (alternative USARSim Robot Interface). This work initially targets the USARSim [33] simulator which was selected against criteria of ease of development and to be very low cost, whilst having high physical and functional fidelity. However altURI could be used with any Windows simulator that uses TCP/IP socket commands and sensor feedback.

USARSim has implemented a robot simulator on a game which has many advantages; games offer some of the most advanced 3D graphics and physics simulations available, provide environment mapping and modeling tools, have efficient multi-user networking and cost very little [19]. Other examples of games based simulators are NERO [31], Faust [7] and SARGE [5].

The altURI middleware is for Windows computers, is fully Open Source under the terms of the BSD License.

II. RELATED WORK

The initial motivation for using middleware was to allow a robot control application to work with both the simulation and a real robot. Moving between virtual and real environments has challenged work in evolutionary robotics for many years [11] [24] due to issues of uncertainty and noise; the so called "reality gap" [13]. These issues have been somewhat mitigated by advances in simulator technology [26] [28].

The use of middleware for real robots is well established and is surveyed by Mohamed et al. [25]. Middleware is also available for simulators and sometimes for both real and simulated environments. Middleware that works with simulators includes: CoRoBa [4], MRDS [23], MOAST [29], OpenRDK [3], Player [8], URBI [2], URSF [15], WURDE [12] and YARP [22]. These examples of middleware cover a very wide range of functions that may be useful when controlling robots and are briefly discussed below.

A. CoRoBa - Controlling Robots with CORBA

CoRoBa works with its own simulator, MoRoS3D, and is a multi-robot framework which controls several robots communicating via CORBA. It initially implemented three robots, their actuators and several sensors. A few high level algorithms such as goal and obstacle navigation have also been implemented. The implementation is strongly standards based and consequently supports extensions to new robots and computing environments well. CoRoba was not considered suitable for this work because it is tied to its simulator (RIDE) and the inter-process communication using CORBA was considered an unnecessary performance (and complexity) overhead when using a single robot.

B. MRDS - Microsoft Robotics Development Studio

MRDS is an integrated robot programming environment with its own simulator (VSE). MRDS consists of service oriented components, a .NET based concurrency library and a visual programming environment using the Visual Studio programming environment. The environment builds on the strengths of its visual programming tools, many robots and sensors are implemented and the support and user base is well resourced. MRDS was not considered suitable because it is not Open Source and is dependent on a single corporate supplier.

C. MOAST - Mobility Open Architecture Simulation and Tools

MOAST works the USARSim simulator and is a design based on the NIST hierarchical 4-D/RCS Reference Model Architecture [1] which hierarchically distributes responsibility for different aspects of robot control and behavior. The implementation enables very complete modeling of the simulator and sensor environment with sophisticated inter-modular communication. The framework also builds on the wide implementation of robots, environments and user base of the USARSim simulator. MOAST was not considered suitable because of the performance (and complexity) overhead of its message passing modular design, it being very highly integrated with USARSim and its dependency on Linux components.

D. OpenRDK - Open Robot Development Kit

OpenRDK is modular framework for controlling robot sensors and actuators that can use several simulators (USARSim, Stage/Gazebo [17], Webots [21] and VSE). OpenRDK implements software processes as agents which instantiate modules that form building block to implement robot systems. OpenRDK was not considered suitable because of its Linux based implementation (users are warned a Windows version is not supported). OpenRDK does include concurrency and information sharing mechanisms but they appear less onerous than other frameworks.

E. Player

The central tool of the Player/Stage/Gazebo (P/S/G) system uses the Stage and Gazebo simulators and is one of the most widely used robot frameworks. Player provides modules to act as servers and subscribers using network communication layers to connect robot and sensor hardware. Many robots and sensors are supported and several higher level algorithms; goal seeking, localization and path planning. The P/S/G system is well supported by active users. In a similar way to MOAST, P/S/G was not considered suitable because of the performance (and complexity) overhead of its message passing modular design and it does not currently run on Windows.

F. URBI - Universal Robotic Body Interface

URBI is a client server based framework that can use the Webots simulator. It includes a low level 'C'-like scripted language for controlling humanoid and animal-like robots via TCP/IP commands. Several robots are supported as server implementations and an extensible object library is also provided. URBI designers emphasize simplicity and it is well supported by active users. URBI was not considered suitable because the Webots simulator is relatively expensive and it geared to humanoid and animal-like robots.

G. URSF - Ubiquitous Robot Simulation Framework

URSF is a framework to control and sense people, appliances, sensors and robots in a virtual simulator and in the real world. The framework builds a world model by interpreting contextual data streams and uses web services standards to communicate between component interfaces. It also provides higher level mapping and path planning features. URSF is a specialized framework tailored to 'ubiquitous' robotic applications so is not considered suitable.

H. WURDE - Washington University Robotics Development Environment

WURDE is a modular tasking and control interface that can use the Stage simulator component of P/S/G. It also contains a control interface for controlling many robots. The WURDE architecture uses four layers of abstraction: Communications, Interfaces, applications, and Systems using message passing inter-process communication. WURDE was not considered suitable because of the performance (and complexity) overhead of its message passing modular design and it does not currently run on Windows.

I. YARP- Yet Another Robot Platform

YARP is middleware that can use the iCub [32] and Webots simulators for controlling humanoid robots implemented as a set of modular processes running on one or more computers. The framework emphasizes code re-use and maintainability and supports inter-process communications, image processing and a class hierarchy. YARP is not considered suitable because of the performance (and complexity) overhead of the message passing modular design and the focus on humanoid robots.

TABLE I. ROBOT MIDDLEWARE COMPARISON

Middleware	Complexity	Support	OS	Open
CoRoBa	High	Low	Win	Yes
MRDS	High	High	Win	No
MOAST	High	High	Win, Linux	Yes
OpenRDK	Medium	High	Unix	Yes
Player	Medium	High	Unix, Mac	Part
URBI	High	Medium	Linux, Mac, Win	Part
URSF	High	Low	Unix	No
WURDE	Medium	Low	Unix	Yes
YARP	Medium	High	Multi	Yes
altURI	Low	?	Win	Yes

The available middleware for use with robot simulators implements sophisticated concurrency mechanisms and sophisticated information sharing mechanisms.

This paper proposes that it is possible to implement a high performance simulator middleware where very simple concurrency and information sharing features are sufficient for real-time control of a vision-based simulator robot.

III. Software Design

The design goals for altURI are derived from the requirements for vision based middleware for robot simulation; high performance, maximal simulator decoupling and running in an easily available environment were:

- To provide modules that are fast, flexible and easily available to as wide as possible range of robot control application development platforms.
- To support image acquisition, robot commands and sensor output for development applications that works with USARSim using a superset of simulator interfaces.

A. *Architecture*

The altURI software contains two components and two support programs. The two components are the vision acquisition module and the command/sensor module.

The vision acquisition module obtains images from the simulator game graphics engine and makes them available to robot control applications. Images are supplied at the same resolution as the simulator but smaller or partial images (multi-view) are available. A command-line support program to start the simulator and load the vision component into a graphics process is provided.

A command/sensor module instance is specific to a robot type (model) and controls a single robot in the simulator. The commands that can be sent and the sensor values that can be received are specified in a configuration file.

The other support component is a test application which loads both modules and can display images retrieved and validate the control and sensor configurations.

Fig. 1 below shows the software modules in context with existing software components of USARSim as well as example controlling application (Matlab)

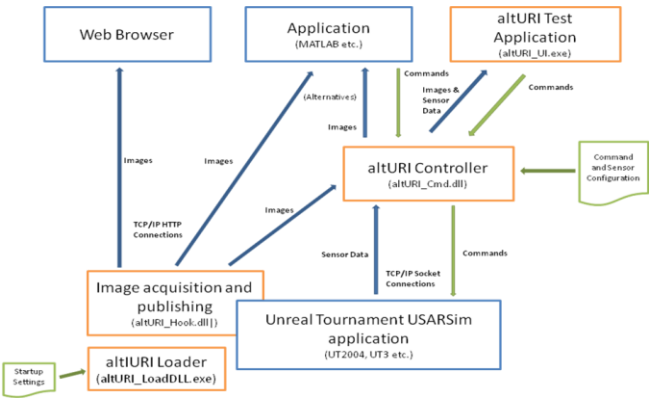

Figure 1. altURI Software Components.

B. *Software Implementation*

The design goals were met by implementing the altURI modules as Windows Dynamic Link Libraries (DLL). The image processing module can supply images from 32-bit and 64-bit graphics applications (i.e. simulators) using DirectX8, DirectX9, DirectX10, DirectX11 or OpenGL graphics for any Windows operating system.

Fig. 2 shows the pseudo code for the hook DLL that captures the images when loaded into a graphics application.

```
graphicsPresent()
1. {
2.   if (frame_requested)
3.     CopyGraphicsBufferToOpenCVImage();
4.     frame_requested = false;      // signal frame available
5.     CallGraphicsPresent();
6. }
```

Figure 2. altURI Host DLL (Graphics Present) Pseudocode.

Fig. 3 shows the pseudo code for the client DLL that requests and receives images from the robot control application.

```
getFrame()
1. {
2.     GetSharedMemoryFrameAddress();
3.     waited=0;
4.     frame_requested  =  true;
5.     while (frame_requested)
6.         wait 1 ms;
7.         waited = waited + 1;
8.         if (waited > 200)          // error timeout
9.             return null;
10.    return frame;                  // from shared memory
11. }
```

Figure 3. altURI Client DLL (GetFrame) Pseudocode.

Images can be supplied as OpenCV [27], USARSim FrameData and Matlab (using the control/sensor module) formats via shared memory or as web images using HTTP requests. The vision component web server can supply images in PNG, JPG or TIFF formats and display a continuous 'movie' of the simulator view.

The altURI modules support a superset of simulator interfaces by working with most graphics engines and by implementing a TCP/IP socket system that can send configured commands with associated numerical values, and retrieve sensor values by parsing returned sensor message text.

C. *Realisation*

The altURI software image acquisition module can retrieve 640x480 pixel images in excess of 30 frames per second from USARSim using a desktop Windows PC. The software has only a small impact on the use of processing

power, about 5% and has a memory footprint of about 10Mb. The software can run on a netbook PC without any issues.

The modules are flexible because they work with USARSim on UT2004 and UT3 but can also acquire images from most other games based environments.

The software works well with USARSim and has been used on projects discussed below. Fig. 4 and Fig. 5 shows the test harness running USARSim displaying captured images. Fig. 6 shows a browser displaying captured images via a network.

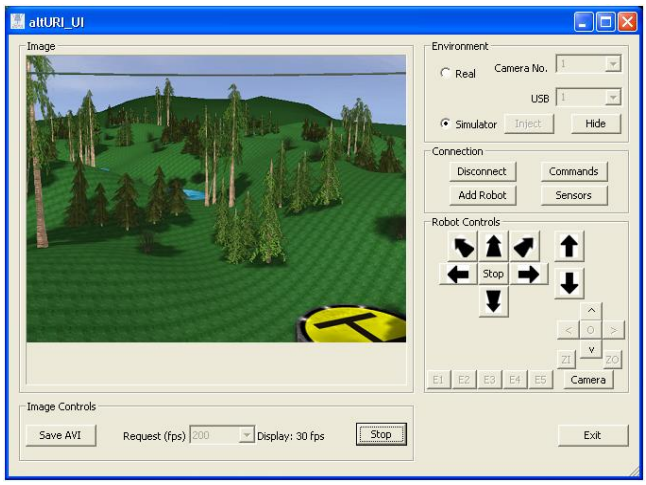

Figure 4. altURI Test Application showing 640x480 pixel image USARSim UT2004 TallTestWorld from AirRobot Camera.

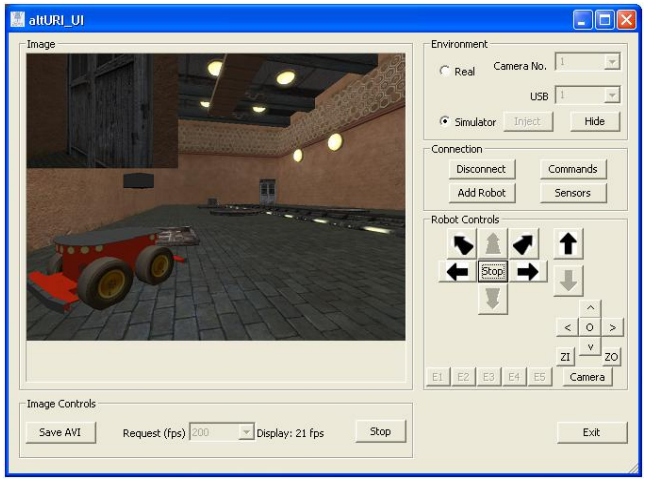

Figure 5. altURI Test Application showing 800x600 pixel image: USARSim UT3 Conveyor map with P3AT Robot.

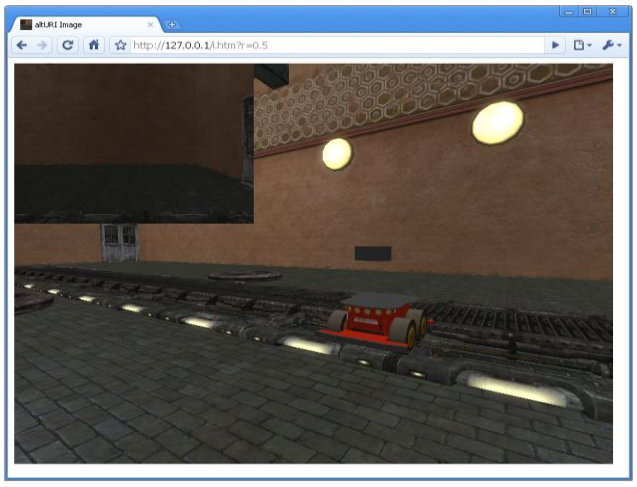

Figure 6. Google Chrome Browser showing 800x600 pixel image, 'movie' refreshing twice every second: USARSim UT3 Conveyor map with P3AT Robot.

IV. Applications

A. *Vision Based Reinforcement Learning*

The altURI system has been used for a study [30] of using a reinforcement learning algorithm for a vision-based approach to landing an Unmanned Aerial Vehicle (UAV). Landing is a difficult challenge for a UAV and the prevalence of accidents during landing justifies the use of a reliable automatic landing system. Difficulties with constructing an accurate model for autonomous control mean that a suitable general learning framework, such as reinforcement learning (RL), is a promising technique.

altURI is used in a visual servoing system where a robot camera keeps track of a visual target while the UAV is steered towards it. An extension of the RL algorithm Least-Squares Policy Iteration [18] is used for the control problem.

The approach was tested with altURI in the USARSim environment with the AirRobot model in an empty NIST Reference Arena [14] using three variations of the algorithm each tested 100 times.

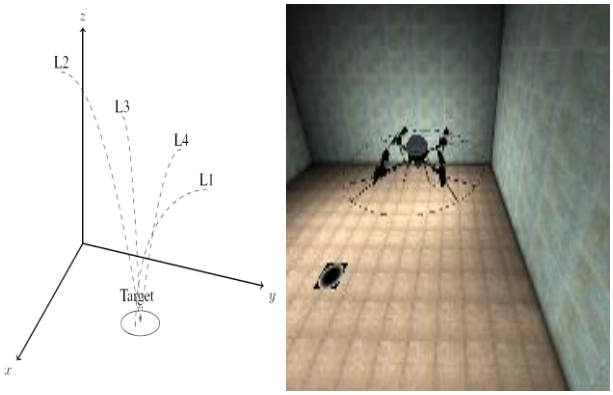

Figure 7. Samples of Learned Paths and USARSim image of the UAV flying toward the target [30].

During this work the middleware providing the images in OpenCV format which worked well with the C++ control application. OpenCV was used for further image processing and this format is useful for ongoing work to transfer the algorithm to the real robot. The middleware was found to be easy to set up and use.

B. Altitude Control Using Elementary Motion Detection

The altURI system is being used for a study using Elementary Motion Detection [35] (EMD) for a vision-based approach to controlling altitude when flying an UAV. Previous work [9] has shown that biologically inspired neural optic flow processing can successfully be used to control aerial robots. However the existing approaches generally use custom neural processing image sensors. This work is investigating other approaches, initially within the USARSim environment.

Motion detection using EMD is calculated using a comparison of two successive images, matrices of point coordinates x and y. The first matrix is used to calculate matrices pixel shifted in the x and y directions respectively (denoted y_{1x}, y_{1y} etc.). Matrices of motion in the x and y directions are:

$$M_x = x_2.*y_{1x} - x_1.*y_{2x}. \quad (1)$$

$$M_y = x_2.*y_{1y} - x_1.*y_{2y}. \quad (2)$$

The magnitude of motion is calculated:

$$M_{xy} = \sqrt{(M_x.^2 + M_y.^2)}. \quad (3)$$

The direction of motion is calculated:

$$M_q = \text{atan2}(M_y, M_x). \quad (4)$$

This work, in its early stages, uses altURI to implement a visual servoing system based on EMD. The servo processing and EMD are written in Matlab.

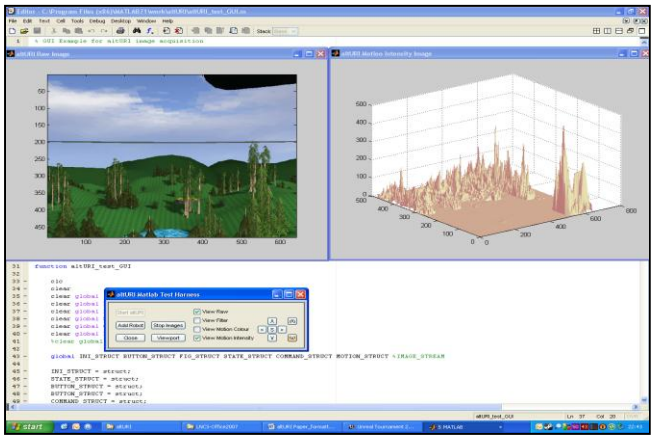

Figure 8. Screenshot: USARSim TallTestWorld, Matlab Real-time 640x480 pixel image and EMD intensity plot for AirRobot UAV camera view.

V. CONCLUSION

In this paper we have described some new middleware for robot simulators that works well with USARSim. By implementing the thinnest possible layer of software the middleware achieves good performance when used for vision-based robot control applications. The software works with most software development environments in Windows and can support other simulators without much effort. The software is very easy to use, it is possible to implement a simple robot control program in less than an hour and add a new robot type in less time.

The altURI software is being used for vision based research now, including a long term research project over the next five years. The software is available for download at: http://webpages.lincoln.ac.uk/mnsmith/altURI.htm. Work will continue to improve and refine the software as a result of user feedback.

REFERENCES

[1] J. Albus, "4-d/rcs reference model architecture for unmanned ground vehicles," in Proceedings IEEE International Conference on Robotics and Automation, pp 3260-3265., 2000.

[2] J. Baillie, "URBI: A universal language for robotic control," in International Journal of Humanoid Robotics, 2004.

[3] D. Calisi, A. Censi, L. Iocchi, D. Nardi, "OpenRDK: a modular framework for robotic software development," in Proceedings of International Conference on Intelligent Robots and Systems (IROS), pp 1872–1877, September 2008.

[4] E. Colon, H. Sahli, Y. Baudoin, "CoRoBa, a multi-mobile robot control and simulation framework," in Special Issue on "Software Development and Integration in Robotics" of the International Journal on Advanced Robotics, Volume 3, Number 1, March 2006, pp 73-78.

[5] J. Craighead, J. Burke, R. Murphy, "Using the Unity Game Engine to Develop SARGE: A Case Study," in Proceedings of International Conference on Intelligent Robots and Systems (IROS), September 2008.

[6] J. Craighead, R. Murphy, J. Burke, B. Goldiez, "A Survey of Commercial & Open Source Unmanned Vehicle Simulators," in Proceedings: ICRA 2007, pp 852-857.

[7] J. Faust, C. Simon, W. D. Smart, "A Video Game-based Mobile Robot Simulation Environment," in Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS'06), Beijing, China 2006.

[8] B.P. Gerkey, R.T. Vaughan, K. Stoy, A. Howard, G.S. Sukhatme, M.J. Mataric, "Most Valuable Player: A Robot Device Server for Distributed Control," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001) pp 1226-1231.

[9] W.E. Green, P.Y. Oh, G. Barrows, "Flying insect inspired vision for autonomous aerial robot manoeuvres in near-earth environments," in Proceedings of IEEE International Conference Robotics & Automation New Orleans. LA. April 2004.

[10] J. Go, B. Browing, M. Veloso, "Accurate and flexible simulation for dynamic, vision-centric robots," in Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04 2004.

[11] I. Harvey, P. Husbands, D.T. Cli, "Issues in evolutionary robotics," in Proceedings of the Second International Conference on Simulation of Adaptive Behaviour, SAB92.

[12] F. Heckel, T. Blakely, M. Dixon, C. Wilson, W.D. Smart, "The WURDE Robotics Middleware and RIDE Multi-Robot Tele-Operation Interface," AAAI Mobile Robot Competition 2006: Papers from the AAAI Workshop, pp 10-14, July 2006.

[13] N. Jakobi, P. Husbands, I. Harvey, “Noise and the reality gap : the use of simulation in evolutionary robotics,” in Lecture Notes in Computer Science 1995.

[14] A. Jacoff, E. Messina, B.A. Weiss, S. Tadokoro, Y. Nakagawa, “Test Arenas and Performance Metrics for Urban Search and Rescue Robots,” in Proceedings of International Conference on Intelligent Robots and Systems IROS 2003.

[15] M. Jang, J. Kim, M. Lee, J. Sohn, “Ubiquitous robot simulation framework and its applications.” In IEEE/RSJ International conference on robots and intelligent systems, pages 3213–8, Edmonton, August 2005.

[16] D. Jung, “OpenSim” http://opensimulator.sourceforge.net/, accessed: 15/03/2011

[17] N. Koening, A. Howard, “Gazebo — 3D Multiple Robot Simulator With Dynamics”, http://playerstage.sourceforge.net/index.php?src=gazebo.

[18] M.G. Lagoudakis, R. Parr, “Least-squares policy iteration,” in Journal of Machine Learning Research, 4:2003.

[19] M. Lewis, J. Jacobson, “Game engines in scientific research,” Communications of the ACM, Volume 45, Number 1, pp 27–31 2002.

[20] O. Michel, “ Khepera Simulator v. 2 User Manual,” University of Nice-Sophia, Antipolis, 1996

[21] O. Michel, “Webots: Symbiosis between virtual and real mobile robots,” in Proceedings of the First International Conference on Virtual Worlds, Paris, France, pp 254-263. Springer Verlag, 1998. See also http://www.cyberbotics.com/webots.

[22] G. Metta, P. Fitzpatrick, L. Natale, “YARP: yet another robot platform,” in International Journal on Advanced Robotics Systems, Vol.3, No. 1, pp 43-48, March 2006.

[23] Microsoft Corporation, Microsoft Robotics Development Studio, http://www.microsoft.com/robotics/ accessed 15/03/2011 (2008)

[24] O. Miglino, H. H. Lund, S. Nolfi, “Evolving Mobile Robots in Simulated and Real Environments,” Artificial Life, 2, pp 417–434. 1996.

[25] N. Mohamed, J. Al-Jaroodi, I. Jawhar, “Middleware for Robotics: A Survey,” in Proceedings. of The IEEE International Conference on Robotics, Automation, and Mechatronics (RAM 2008), pp 736-742.

[26] S. Okamoto, K. Kurose, S. Saga, K. Ohno, S. Tadokoro, “Validation of simulated robots with realistically modelled dimensions and mass in USARsim,” in Safety, Security and Rescue Robotics, SSRR 2008. IEEE International Workshop on, Oct. 2008, pp 77–82.

[27] OpenCV, http://opencv.willowgarage.com/wiki/ acessed 15/03/2011.

[28] C. Pepper, S. Balakirsky, C. Scrapper, “Robot Simulation Physics Validation,” in Proceedings of PerMIS’07.

[29] C. Scrapper, S. Balakirsky, E. Messina, “MOAST and USARSim - A Combined Framework for the Development and Testing of Autonomous Systems,” in Proceedings of the SPIE Defense and Security Symposium 2006.

[30] M. Shaker, M. N. R. Smith, S. Yue, T. Duckett, "Vision-based landing of a simulated unmanned aerial vehicle with fast reinforcement learning," in International Symposium on Learning and Adaptive Behaviour in Robotics Systems (LAB-RS 2010), 6-7 September 2010, Canterbury, UK.

[31] K. Stanley, B. Bryant, R. Miikkulainen, “Evolving Neural Network Agents in the NERO Video Game,” in Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG’05). Piscataway, NJ.

[32] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G Metta, L Natale, F. Nori “An open-source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator,” in Proceedings of IEEE workshop on performance metrics for intelligent systems workshop, 2008.

[33] J. Wang, M. Lewis, J. Gennari, “USAR: A game based simulation for teleoperation,” in Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society 2003.

[34] J. Wang, M. Lewis, S Hughes, M Koes, S Carpin, “Validating USARsim for use in HRI research,” in: Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting, pp 457–461 2005.

[35] J. M. Zanker, “On the elementary mechanism underlying secondary motion processing,” Phil. Trans. R. Soc. London B 351: 1725-1736 1996.